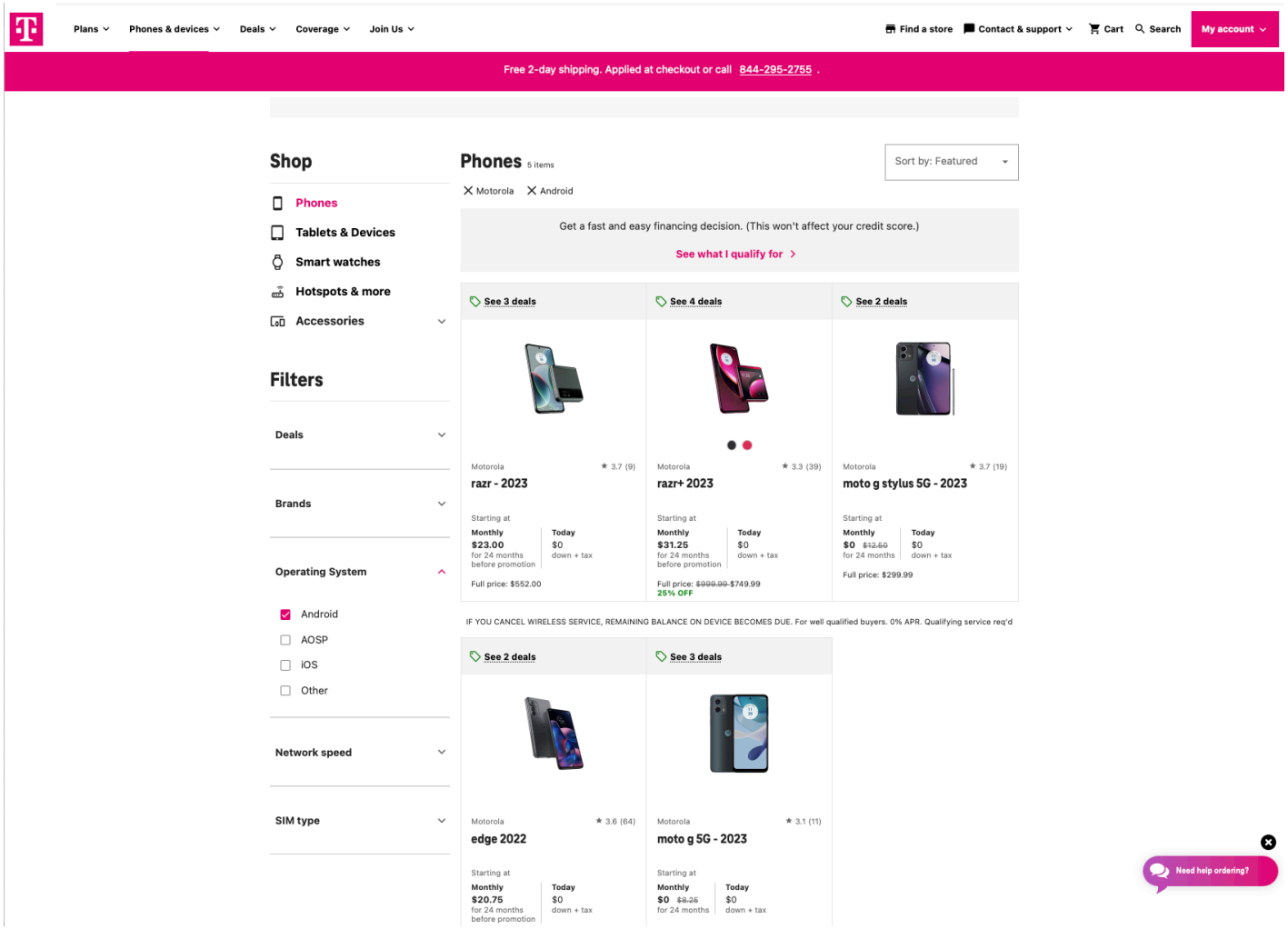
EXHIBIT 7

Exhibit C - U.S. Patent No. 8,589,541 ("'541 Patent")

Accused Instrumentalities: smartphones, basic phones, tablets, laptops, and hotspot devices sold (including those sold in bundles with data plans) or used by T-Mobile and all versions and variations thereof ("Accused Instrumentalities") since the issuance of U.S. Pat. No. 8,589,541 (the "Asserted Patent").

Claim 1

Claim	Public Documentation
[1a] A non-transitory computer-readable storage medium storing machine-executable instructions that, when executed by one or more processors of a wireless end-user device, cause the one or more processors to:	The Accused Instrumentalities include "A non-transitory computer-readable storage medium storing machine-executable instructions that, when executed by one or more processors of a wireless end-user device, cause the one or more processors to." For example, T-Mobile sells and uses devices described by T-Mobile's website below (e.g., devices made by Samsung, Apple, Motorola, Google, Nokia, etc.). These devices constitute a wireless end-user device as described in claim 1. *See, e.g.* https://www.t-mobile.com/cell-phones

Claim	Public Documentation
	 ; *see also* https://www.t-mobile.com/tablets; https://www.t-mobile.com/smart-watches; https://www.t-mobile.com/hotspots-iot-connected-devices.

Claim	Public Documentation
	As a specific example, Motorola's devices, including the Motorola razr+ 2023, are wireless end-user devices which run the Android Operating System, and include a processor. *See, e.g.*, https://www.t-mobile.com/cell-phone/motorola-razr-plus-2023:

Additional spec details

Battery Description	3800 mAh
Ports	USB Type-C
Connectivity	Wi-Fi 802.11 a/b/g/n/ac/k/v/r/ax/Wi-Fi 6e, Bluetooth® 5.3, NFC
Processor	Qualcomm of Snapdragon 8+ Gen 1 Mobile Platform
Operating System	Android
Ram	8 GB
Maximum Expandable Memory	0 GB
Wireless Network Technology Generations	4G LTE, 5G
Supported Email Platforms	Apple Mail, POP3, IMAP4, SMTP, Microsoft® Exchange, AOL, AIM, Yahoo!® Mail, GMail
Hearing Aid Compatibility	M3, T3
WEA Capable	true
Mobile Hotspot Capable	true
Frequency	GSM: 850 MHz, 900 MHz, 1800 MHz, 1900 MHz; UMTS: Band I (2100), Band II (1900), Band IV (1700/2100), Band VIII (900); 5G: n2, n5, n7, n12, n13, n14, n25, n26, n29, n30, n38, n41, n48, n66, n70, n71, n77, n78; LTE: 2, 3, 4, 5, 7, 8, 12, 13, 14, 17, 18, 19, 20, 25, 26, 28, 29, 30, 38, 39, 40, 41, 46, 48, 66, 71
Weight	6.56 Ounces
Length	0.28
Height	6.73
Width	2.91

What's in the box

- Motorola razr+ 2023
- USB Type-C Cable
- Guides
- SIM Tool

For WEA capability, see T-Mobile WEA

California residents: see the California Proposition 65 **WARNING**

Claim	Public Documentation
	For further example, the Motorola razr+ 2023 model is sold or used by T-Mobile and includes 256GB of storage, in which control policies for applications are stored. *See, e.g.*, *id.*

[1b] identify a service usage activity of the wireless end-user device, the service usage activity being associated with a first software component of a plurality of software components on the wireless end-user device, the service usage activity comprising one or more prospective or successful communications over a wireless network;	The Accused Instrumentalities "identify a service usage activity of the wireless end-user device, the service usage activity being associated with a first software component of a plurality of software components on the wireless end-user device, the service usage activity comprising one or more prospective or successful communications over a wireless network." For example, Motorola's devices, including the Motorola razr+ 2023, run the Android Operating System, which includes features such as "Data Saver," "Battery Saver," "Extreme Battery Saver," "Doze Mode," "App Standby," "Adaptive Battery," and/or "JobScheduler" which apply to at least some service usage activities associated with a software component comprising prospective or successful communications over a wireless network e.g., when apps utilize network access, jobs, syncs, alarms, etc. *See, e.g.*, https://www.t-mobile.com/cell-phone/motorola-razr-plus-2023:

Additional spec details

Battery Description	3800 mAh
Ports	USB Type-C
Connectivity	Wi-Fi 802.11 a/b/g/n/ac/k/v/r/ax/Wi-Fi 6e, Bluetooth® 5.3, NFC
Processor	Qualcomm of Snapdragon 8+ Gen 1 Mobile Platform
Operating System	Android
Ram	8 GB
Maximum Expandable Memory	0 GB
Wireless Network Technology Generations	4G LTE, 5G
Supported Email Platforms	Apple Mail, POP3, IMAP4, SMTP, Microsoft® Exchange, AOL, AIM, Yahoo!® Mail, GMail
Hearing Aid Compatibility	M3, T3
WEA Capable	true
Mobile Hotspot Capable	true
Frequency	GSM: 850 MHz, 900 MHz, 1800 MHz, 1900 MHz; UMTS: Band I (2100), Band II (1900), Band IV (1700/2100), Band VIII (900); 5G: n2, n5, n7, n12, n13, n14, n25, n26, n29, n30, n38, n41, n48, n66, n70, n71, n77, n78; LTE: 2, 3, 4, 5, 7, 8, 12, 13, 14, 17, 18, 19, 20, 25, 26, 28, 29, 30, 38, 39, 40, 41, 46, 48, 66, 71
Weight	6.56 Ounces
Length	0.28
Height	6.73
Width	2.91

What's in the box

- Motorola razr+ 2023
- USB Type-C Cable
- Guides
- SIM Tool

For WEA capability, see T-Mobile WEA

California residents: see the California Proposition 65 **WARNING**

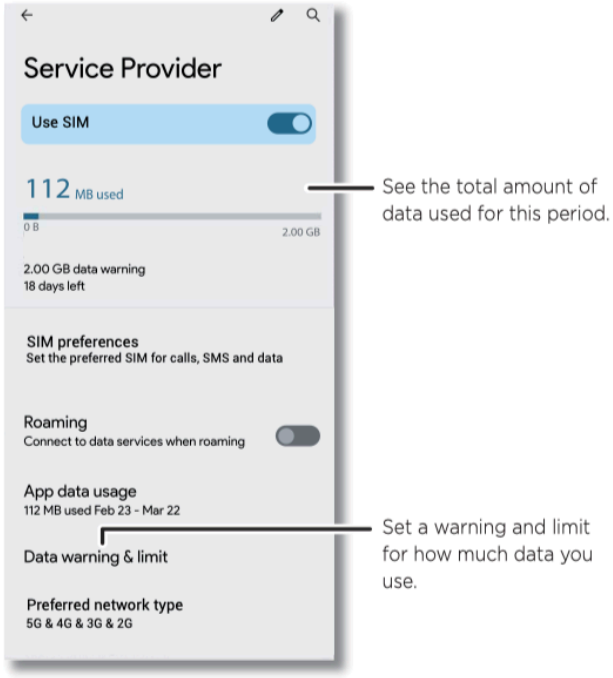
; https://en-us.support.motorola.com/ci/fattach/get/81413275/1686837226/redirect/1/session/L2F2LzEvdGltZS8xNzEwNzI3MjU3L2dlbi8xNzEwNzI3MjU3L3NpZC9mVWc0dzJqVlJvVU92V2lZJTdFNGNwU3JySkJMa-VJnd2hnMUtYaDdrXzZQYUQ3dVZrSDlmaHNOU05ZSWFWJTdFb2F1Wng4OUpNTEFsWU50WHF1aE

	w0YWFSOTc2M3NWUGRBeEdPRXVyc1A1UGJQVHZmYXhiWEw2dzg2UHdnJTIxJTIx/file-name/motorola+razr%2B+-+2023.NA+Retail.UG.en-US.SSC8D89424-B.pdf:

Data usage

You can track the amount of data your phone uploads and downloads.

Find it: Swipe up from the home screen and tap **Settings** > **Network & internet** > **Mobile network** > choose a SIM card



To see a breakdown of which apps are using data, swipe up from the home screen and tap **Settings** > **Network & internet** > **Mobile network** > **App data usage**. Some apps transfer data in the background when you're not viewing them—to help reduce this type of data usage, swipe up from the home screen and tap **Settings** > **Network & internet** > **Data Saver**, then tap the switch next to **Use Data Saver** to turn it on.

Tip: To see Wi-FI data usage, swipe up from the home screen and tap **Settings** > **Network & internet** > **Internet** > **Non-carrier data usage**.

Note: Usage information is provided to help you manage your phone. This may not match the amounts charged by your service provider, as they're not measured in the same way.

Improve battery life

Your phone processes tons of information. Depending on what apps are in use, your phone may use a lot of power.

When your phone is not in use for a period of time, unnecessary background processes are shut down to optimize battery life.

» To see what's using up battery power, swipe up from the home screen and tap **Settings** > **Battery** > **Battery usage**.

» To help improve battery life, swipe up from the home screen and tap **Settings** > **Battery** > **Battery Saver**, and tap the switch next to **Use Battery Saver** to turn it on. When on, your phone's display changes to Dark theme.

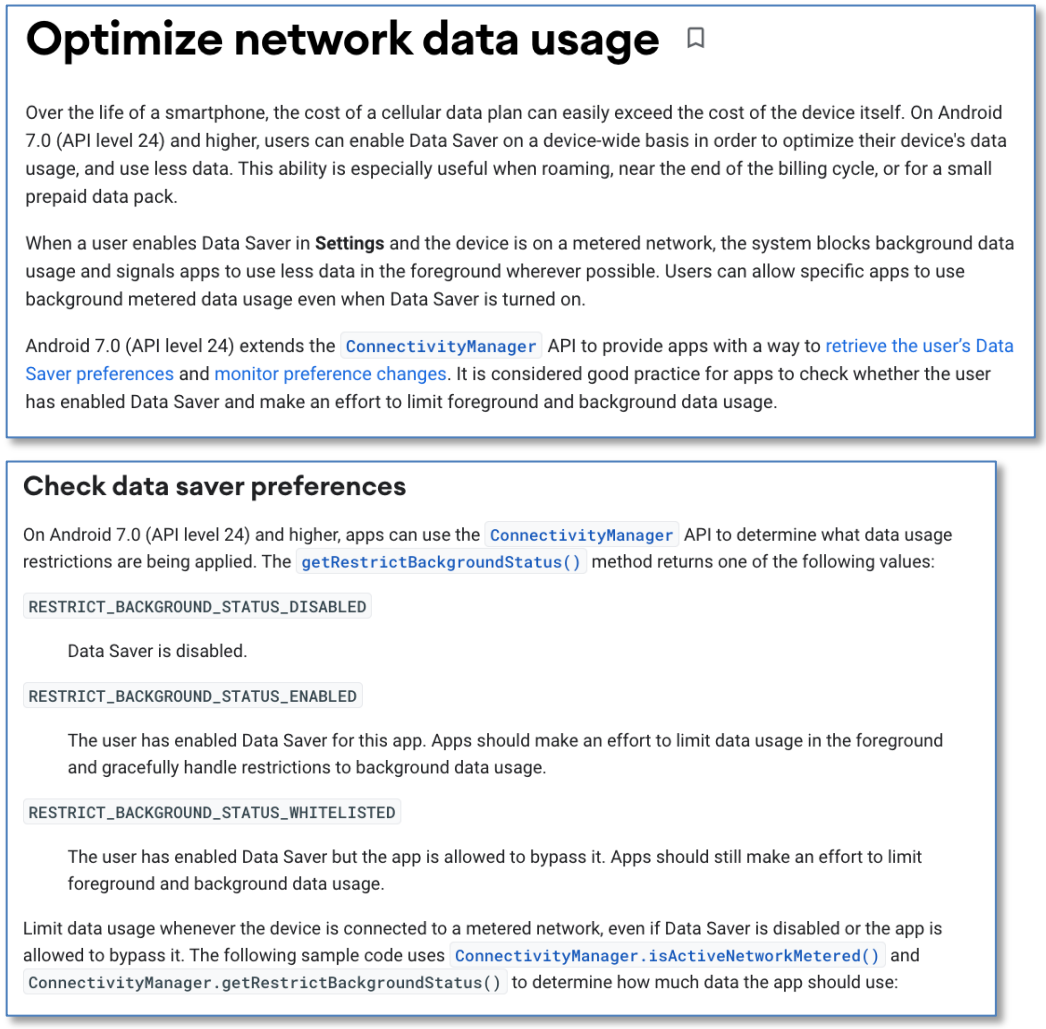
» To limit battery use for apps that you don't use often, swipe up from the home screen and tap **Settings** > **Battery** > **Adaptive Battery**, and turn it on.

» To charge more efficiently and keep your battery healthy, swipe up from the home screen and tap **Settings** > **Battery** > **Optimized charging**, and turn it on.

» To protect your battery from being overcharged, swipe up from the home screen and tap **Settings** > **Battery** > **Overcharge protection**, and turn it on.

» To show battery percentage in status bar, swipe up from the home screen and tap **Settings** > **Battery**, then tap the switch next to **Battery Percentage** to turn it on.

; https://developer.android.com/training/basics/network-ops/data-saver:



Optimize network data usage

Over the life of a smartphone, the cost of a cellular data plan can easily exceed the cost of the device itself. On Android 7.0 (API level 24) and higher, users can enable Data Saver on a device-wide basis in order to optimize their device's data usage, and use less data. This ability is especially useful when roaming, near the end of the billing cycle, or for a small prepaid data pack.

When a user enables Data Saver in **Settings** and the device is on a metered network, the system blocks background data usage and signals apps to use less data in the foreground wherever possible. Users can allow specific apps to use background metered data usage even when Data Saver is turned on.

Android 7.0 (API level 24) extends the `ConnectivityManager` API to provide apps with a way to retrieve the user's Data Saver preferences and monitor preference changes. It is considered good practice for apps to check whether the user has enabled Data Saver and make an effort to limit foreground and background data usage.

Check data saver preferences

On Android 7.0 (API level 24) and higher, apps can use the `ConnectivityManager` API to determine what data usage restrictions are being applied. The `getRestrictBackgroundStatus()` method returns one of the following values:

`RESTRICT_BACKGROUND_STATUS_DISABLED`

Data Saver is disabled.

`RESTRICT_BACKGROUND_STATUS_ENABLED`

The user has enabled Data Saver for this app. Apps should make an effort to limit data usage in the foreground and gracefully handle restrictions to background data usage.

`RESTRICT_BACKGROUND_STATUS_WHITELISTED`

The user has enabled Data Saver but the app is allowed to bypass it. Apps should still make an effort to limit foreground and background data usage.

Limit data usage whenever the device is connected to a metered network, even if Data Saver is disabled or the app is allowed to bypass it. The following sample code uses `ConnectivityManager.isActiveNetworkMetered()` and `ConnectivityManager.getRestrictBackgroundStatus()` to determine how much data the app should use:

; https://developer.android.com/training/monitoring-device-state/doze-standby:

Optimize for Doze and App Standby

Starting from Android 6.0 (API level 23), Android introduces two power-saving features that extend battery life for users by managing how apps behave when a device is not connected to a power source. *Doze* reduces battery consumption by deferring background CPU and network activity for apps when the device is unused for long periods of time. *App Standby* defers background network activity for apps with which the user has not recently interacted.

While the device is in Doze, apps' access to certain battery-intensive resources is deferred until maintenance windows. The specific restrictions are listed in Power Management Restrictions.

Doze and App Standby manage the behavior of all apps running on Android 6.0 or higher, regardless whether they are specifically targeting API level 23. To ensure the best experience for users, test your app in Doze and App Standby modes and make any necessary adjustments to your code. The sections below provide details.

Understanding Doze

If a user leaves a device unplugged and stationary for a period of time, with the screen off, the device enters Doze mode. In Doze mode, the system attempts to conserve battery by restricting apps' access to network and CPU-intensive services. It also prevents apps from accessing the network and defers their jobs, syncs, and standard alarms.

Periodically, the system exits Doze for a brief time to let apps complete their deferred activities. During this *maintenance window*, the system runs all pending syncs, jobs, and alarms, and lets apps access the network.

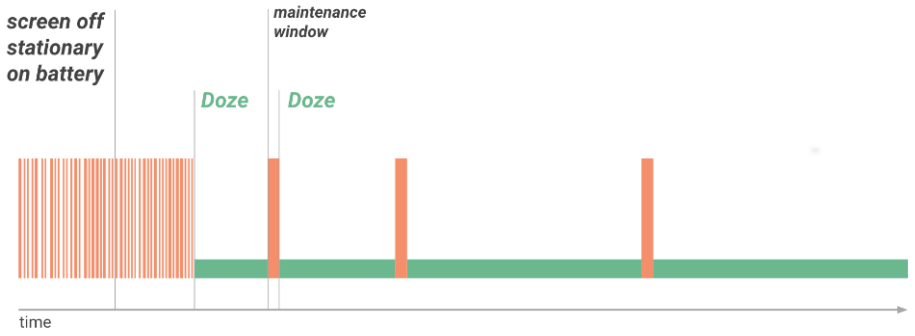


Figure 1. Doze provides a recurring maintenance window for apps to use the network and handle pending activities.

	At the conclusion of each maintenance window, the system again enters Doze, suspending network access and deferring jobs, syncs, and alarms. Over time, the system schedules maintenance windows less and less frequently, helping to reduce battery consumption in cases of longer-term inactivity when the device is not connected to a charger. As soon as the user wakes the device by moving it, turning on the screen, or connecting a charger, the system exits Doze and all apps return to normal activity. The Doze restriction on network access is also likely to affect your app, especially if the app relies on real-time messages such as tickles or notifications. If your app requires a persistent connection to the network to receive messages, you should use Firebase Cloud Messaging (FCM) if possible. ; https://developer.android.com/topic/performance/appstandby:

App Standby Buckets

Android 9 (API level 28) and higher support **App Standby Buckets**. App Standby Buckets help the system prioritize apps' requests for resources based on how recently and how frequently the apps are used. Based on app usage patterns, each app is placed in one of five priority **buckets**. The system limits the device resources available to each app based on which bucket the app is in.

Priority buckets

The system dynamically assigns each app to a priority bucket, reassigning the apps as needed. The system may rely on a preloaded app that uses machine learning to determine how likely each app is to be used, and assigns apps to the appropriate buckets. If the system app is not present on a device, the system defaults to sorting apps based on how recently they were used. More active apps are assigned to buckets that give the apps higher priority, making more system resources available to the app. In particular, the bucket determines how frequently the app's jobs run, and how often the app can trigger alarms. These restrictions apply only while the device is on battery power; the system does not impose these restrictions on apps while the device is charging.

Note: Every manufacturer can set their own criteria for how non-active apps are assigned to buckets. You should not try to influence which bucket your app is assigned to. Instead, focus on making sure your app behaves well in whatever bucket it might be in. Your app can find out what bucket it's currently in by calling `UsageStatsManager.getAppStandbyBucket()`.

The buckets are:

1. Active: App is currently being used or was very recently used.
2. Working set: App is in regular use.
3. Frequent: App is often used, but not every day.
4. Rare: App is not frequently used.
5. Restricted: App consumes a great deal of system resources, or may exhibit undesirable behavior.

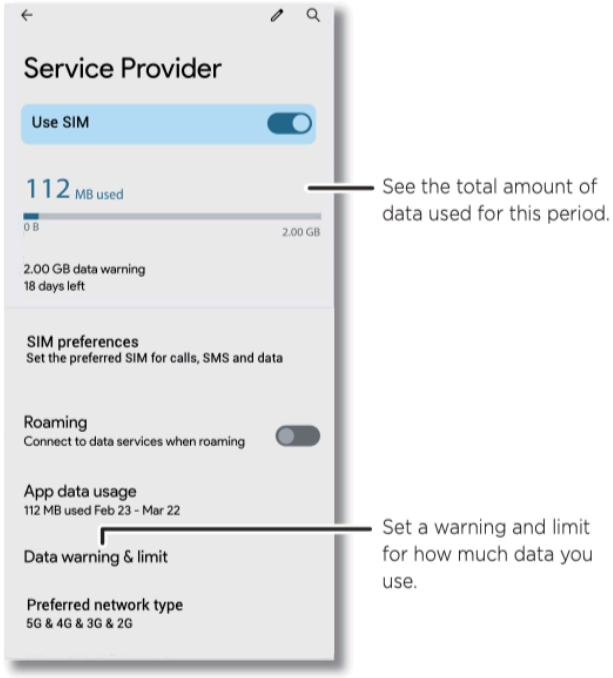
In addition, there's a special **never** bucket for apps that have been installed but have never been run. The system imposes severe restrictions on these apps.

	; https://developer.android.com/topic/performance/background-optimization; https://developer.android.com/reference/android/app/job/JobScheduler; https://developer.android.com/guide/background/persistent; https://developer.android.com/guide/components/services; https://developer.android.com/guide/components/activities/intro-activities; https://developer.android.com/reference/java/net/URLConnection; https://developer.android.com/training/articles/security-ssl; https://developer.android.com/reference/android/net/DnsResolver; https://developer.android.com/guide/topics/media; https://developer.android.com/media; https://developer.android.com/guide/topics/media/platform/mediaplayer.
[1c] determine whether the service usage activity comprises a background activity;	The Accused Instrumentalities "determine whether the service usage activity comprises a background activity." For example, Motorola devices determine whether the service usage activity comprises background or foreground activity over wireless networks, e.g., when apps utilize network access, jobs, syncs, alarms, etc. *See, e.g.*, https://en-us.support.motorola.com/ci/fattach/get/81413275/1686837226/redirect/1/session/L2F2LzEvdGltZS8xNzEwNzI3MjU3L2dlbi8xNzEwNzI3MjU3L3NpZC9mVWc0dzJqVlJvVU92V2lZJTdFNGNwU3JySkJMa-VJnd2hnMUtYaDdrXzZQYUQ3dVZrSDlmaHNOU05ZSWFWJTdFb2F1Wng4OUpNTEFsWU50WHF1aEw0YWFSOTc2M3NWUGRBeEdPRXVyc1A1UGJQVHZmYXhiWEw2dzg2UHdnJTIxJTIx/filename/motorola+razr%2B+-+2023.NA+Retail.UG.en-US.SSC8D89424-B.pdf:

Data usage

You can track the amount of data your phone uploads and downloads.

Find it: Swipe up from the home screen and tap **Settings** > **Network & internet** > **Mobile network** > choose a SIM card



To see a breakdown of which apps are using data, swipe up from the home screen and tap **Settings** > **Network & internet** > **Mobile network** > **App data usage**. Some apps transfer data in the background when you're not viewing them—to help reduce this type of data usage, swipe up from the home screen and tap **Settings** > **Network & internet** > **Data Saver**, then tap the switch next to **Use Data Saver** to turn it on.

Tip: To see Wi-FI data usage, swipe up from the home screen and tap **Settings** > **Network & internet** > **Internet** > **Non-carrier data usage**.

Note: Usage information is provided to help you manage your phone. This may not match the amounts charged by your service provider, as they're not measured in the same way.

Improve battery life

Your phone processes tons of information. Depending on what apps are in use, your phone may use a lot of power.

When your phone is not in use for a period of time, unnecessary background processes are shut down to optimize battery life.

» To see what's using up battery power, swipe up from the home screen and tap **Settings** > **Battery** > **Battery usage**.

» To help improve battery life, swipe up from the home screen and tap **Settings** > **Battery** > **Battery Saver**, and tap the switch next to **Use Battery Saver** to turn it on. When on, your phone's display changes to Dark theme.

» To limit battery use for apps that you don't use often, swipe up from the home screen and tap **Settings** > **Battery** > **Adaptive Battery**, and turn it on.

» To charge more efficiently and keep your battery healthy, swipe up from the home screen and tap **Settings** > **Battery** > **Optimized charging**, and turn it on.

» To protect your battery from being overcharged, swipe up from the home screen and tap **Settings** > **Battery** > **Overcharge protection**, and turn it on.

» To show battery percentage in status bar, swipe up from the home screen and tap **Settings** > **Battery**, then tap the switch next to **Battery Percentage** to turn it on.

; https://developer.android.com/training/basics/network-ops/data-saver:

Optimize network data usage

Over the life of a smartphone, the cost of a cellular data plan can easily exceed the cost of the device itself. On Android 7.0 (API level 24) and higher, users can enable Data Saver on a device-wide basis in order to optimize their device's data usage, and use less data. This ability is especially useful when roaming, near the end of the billing cycle, or for a small prepaid data pack.

When a user enables Data Saver in **Settings** and the device is on a metered network, the system blocks background data usage and signals apps to use less data in the foreground wherever possible. Users can allow specific apps to use background metered data usage even when Data Saver is turned on.

Android 7.0 (API level 24) extends the `ConnectivityManager` API to provide apps with a way to retrieve the user's Data Saver preferences and monitor preference changes. It is considered good practice for apps to check whether the user has enabled Data Saver and make an effort to limit foreground and background data usage.

Check data saver preferences

On Android 7.0 (API level 24) and higher, apps can use the `ConnectivityManager` API to determine what data usage restrictions are being applied. The `getRestrictBackgroundStatus()` method returns one of the following values:

`RESTRICT_BACKGROUND_STATUS_DISABLED`

Data Saver is disabled.

`RESTRICT_BACKGROUND_STATUS_ENABLED`

The user has enabled Data Saver for this app. Apps should make an effort to limit data usage in the foreground and gracefully handle restrictions to background data usage.

`RESTRICT_BACKGROUND_STATUS_WHITELISTED`

The user has enabled Data Saver but the app is allowed to bypass it. Apps should still make an effort to limit foreground and background data usage.

Limit data usage whenever the device is connected to a metered network, even if Data Saver is disabled or the app is allowed to bypass it. The following sample code uses `ConnectivityManager.isActiveNetworkMetered()` and `ConnectivityManager.getRestrictBackgroundStatus()` to determine how much data the app should use:

; https://developer.android.com/training/monitoring-device-state/doze-standby:

Optimize for Doze and App Standby

Starting from Android 6.0 (API level 23), Android introduces two power-saving features that extend battery life for users by managing how apps behave when a device is not connected to a power source. *Doze* reduces battery consumption by deferring background CPU and network activity for apps when the device is unused for long periods of time. *App Standby* defers background network activity for apps with which the user has not recently interacted.

While the device is in Doze, apps' access to certain battery-intensive resources is deferred until maintenance windows. The specific restrictions are listed in Power Management Restrictions.

Doze and App Standby manage the behavior of all apps running on Android 6.0 or higher, regardless whether they are specifically targeting API level 23. To ensure the best experience for users, test your app in Doze and App Standby modes and make any necessary adjustments to your code. The sections below provide details.

Understanding Doze

If a user leaves a device unplugged and stationary for a period of time, with the screen off, the device enters Doze mode. In Doze mode, the system attempts to conserve battery by restricting apps' access to network and CPU-intensive services. It also prevents apps from accessing the network and defers their jobs, syncs, and standard alarms.

Periodically, the system exits Doze for a brief time to let apps complete their deferred activities. During this *maintenance window*, the system runs all pending syncs, jobs, and alarms, and lets apps access the network.

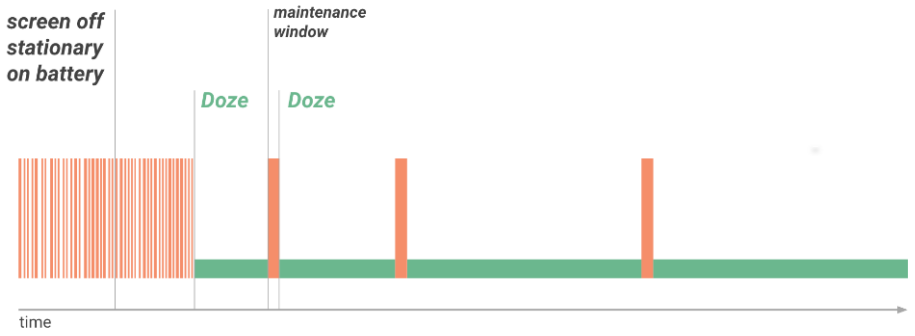


Figure 1. Doze provides a recurring maintenance window for apps to use the network and handle pending activities.

	At the conclusion of each maintenance window, the system again enters Doze, suspending network access and deferring jobs, syncs, and alarms. Over time, the system schedules maintenance windows less and less frequently, helping to reduce battery consumption in cases of longer-term inactivity when the device is not connected to a charger. As soon as the user wakes the device by moving it, turning on the screen, or connecting a charger, the system exits Doze and all apps return to normal activity. The Doze restriction on network access is also likely to affect your app, especially if the app relies on real-time messages such as tickles or notifications. If your app requires a persistent connection to the network to receive messages, you should use Firebase Cloud Messaging (FCM) if possible. ; https://developer.android.com/topic/performance/appstandby:

App Standby Buckets

Android 9 (API level 28) and higher support **App Standby Buckets**. App Standby Buckets help the system prioritize apps' requests for resources based on how recently and how frequently the apps are used. Based on app usage patterns, each app is placed in one of five priority **buckets**. The system limits the device resources available to each app based on which bucket the app is in.

Priority buckets

The system dynamically assigns each app to a priority bucket, reassigning the apps as needed. The system may rely on a preloaded app that uses machine learning to determine how likely each app is to be used, and assigns apps to the appropriate buckets. If the system app is not present on a device, the system defaults to sorting apps based on how recently they were used. More active apps are assigned to buckets that give the apps higher priority, making more system resources available to the app. In particular, the bucket determines how frequently the app's jobs run, and how often the app can trigger alarms. These restrictions apply only while the device is on battery power; the system does not impose these restrictions on apps while the device is charging.

Note: Every manufacturer can set their own criteria for how non-active apps are assigned to buckets. You should not try to influence which bucket your app is assigned to. Instead, focus on making sure your app behaves well in whatever bucket it might be in. Your app can find out what bucket it's currently in by calling `UsageStatsManager.getAppStandbyBucket()`.

The buckets are:

1. Active: App is currently being used or was very recently used.
2. Working set: App is in regular use.
3. Frequent: App is often used, but not every day.
4. Rare: App is not frequently used.
5. Restricted: App consumes a great deal of system resources, or may exhibit undesirable behavior.

In addition, there's a special **never** bucket for apps that have been installed but have never been run. The system imposes severe restrictions on these apps.

; https://developer.android.com/topic/performance/power/power-details; https://developer.android.com/topic/performance/background-optimization; https://developer.android.com/reference/android/app/job/JobScheduler; https://developer.android.com/guide/background/persistent; https://developer.android.com/guide/components/activities/activity-lifecycle; https://developer.android.com/guide/components/activities/process-lifecycle:

1. A **foreground process** is one that is required for what the user is currently doing. Various application components can cause its containing process to be considered foreground in different ways. A process is considered to be in the foreground if any of the following conditions hold:
   - It is running an `Activity` at the top of the screen that the user is interacting with (its `onResume()` method has been called).
   - It has a `BroadcastReceiver` that is currently running (its `BroadcastReceiver.onReceive()` method is executing).
   - It has a `Service` that is currently executing code in one of its callbacks (`Service.onCreate()`, `Service.onStart()`, or `Service.onDestroy()`).

   There will only ever be a few such processes in the system, and these will only be killed as a last resort if memory is so low that not even these processes can continue to run. Generally, at this point, the device has reached a memory paging state, so this action is required in order to keep the user interface responsive.

; https://developer.android.com/guide/background:

	Definition of background work An app is running in the *background* when both the following conditions are satisfied: • None of the app's activities are currently visible to the user. • The app isn't running any foreground services that started while an activity from the app was visible to the user. Otherwise, the app is running in the *foreground*. ; https://developer.android.com/guide/components/services;

Types of Services

These are the three different types of services:

Foreground

A foreground service performs some operation that is noticeable to the user. For example, an audio app would use a foreground service to play an audio track. Foreground services must display a Notification. Foreground services continue running even when the user isn't interacting with the app.

When you use a foreground service, you must display a notification so that users are actively aware that the service is running. This notification cannot be dismissed unless the service is either stopped or removed from the foreground.

Learn more about how to configure foreground services in your app.

> **Note:** The WorkManager API offers a flexible way of scheduling tasks, and is able to run these jobs as foreground services if needed. In many cases, using WorkManager is preferable to using foreground services directly.

Background

A background service performs an operation that isn't directly noticed by the user. For example, if an app used a service to compact its storage, that would usually be a background service.

> **Note:** If your app targets API level 26 or higher, the system imposes restrictions on running background services when the app itself isn't in the foreground. In most situations, for example, you shouldn't access location information from the background. Instead, schedule tasks using WorkManager.

Bound

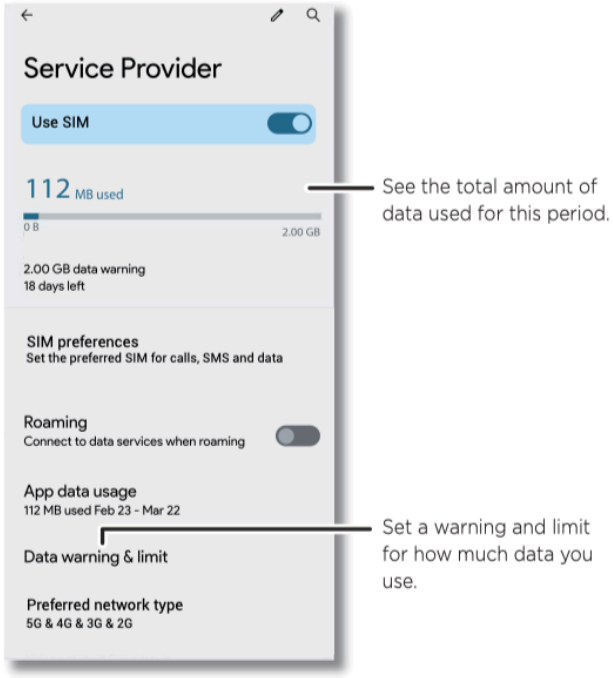
A service is *bound* when an application component binds to it by calling `bindService()`. A bound service offers a client-server interface that allows components to interact with the service, send requests, receive results, and even do so across processes with interprocess communication (IPC). A bound service runs only as long as another application component is bound to it. Multiple components can bind to the service at once, but when all of them unbind, the service is destroyed.

	; https://developer.android.com/guide/components/activities/intro-activities.
[1d] determine at least an aspect of a policy based on a user input obtained through a user interface of the wireless end-user device or based on information from a network element, the policy to be applied if the service usage activity is the background activity, the policy at least for controlling the service usage activity;	The Accused Instrumentalities "determine at least an aspect of a policy based on a user input obtained through a user interface of the wireless end-user device or based on information from a network element, the policy to be applied if the service usage activity is the background activity, the policy at least for controlling the service usage activity." For example, Motorola devices include an interface which allow users to specify multiple aspects of policies based on user input in various settings (e.g., enabling/disabling Data Saver, Power Saver, Adaptive Battery, Doze features, as well as enabling/disabling policies for specific applications) for controlling service usage activities over wireless networks, e.g., when apps utilize network access, jobs, syncs, alarms, etc. *See, e.g.*, https://en-us.support.motorola.com/ci/fattach/get/81413275/1686837226/redirect/1/session/L2F2LzEvdGltZS8xNzEwNzI3MjU3L2dlbi8xNzEwNzI3MjU3L3NpZC9mVWc0dzJqVlJvVU92V2lZJTdFNGNwU3JySkJMa-VJnd2hnMUtYaDdrXzZQYUQ3dVZrSDlmaHNOU05ZSWFWJTdFb2F1Wng4OUpNTEFsWU50WHF1aEw0YWFSOTc2M3NWUGRBeEdPRXVyc1A1UGJQVHZmYXhiWEw2dzg2UHdnJTIxJTIx/filename/motorola+razr%2B+-+2023.NA+Retail.UG.en-US.SSC8D89424-B.pdf:

Data usage

You can track the amount of data your phone uploads and downloads.

Find it: Swipe up from the home screen and tap **Settings** > **Network & internet** > **Mobile network** > choose a SIM card



To see a breakdown of which apps are using data, swipe up from the home screen and tap **Settings** > **Network & internet** > **Mobile network** > **App data usage**. Some apps transfer data in the background when you're not viewing them—to help reduce this type of data usage, swipe up from the home screen and tap **Settings** > **Network & internet** > **Data Saver**, then tap the switch next to **Use Data Saver** to turn it on.

Tip: To see Wi-FI data usage, swipe up from the home screen and tap **Settings** > **Network & internet** > **Internet** > **Non-carrier data usage**.

Note: Usage information is provided to help you manage your phone. This may not match the amounts charged by your service provider, as they're not measured in the same way.

Improve battery life

Your phone processes tons of information. Depending on what apps are in use, your phone may use a lot of power.

When your phone is not in use for a period of time, unnecessary background processes are shut down to optimize battery life.

» To see what's using up battery power, swipe up from the home screen and tap **Settings** > **Battery** > **Battery usage**.

» To help improve battery life, swipe up from the home screen and tap **Settings** > **Battery** > **Battery Saver**, and tap the switch next to **Use Battery Saver** to turn it on. When on, your phone's display changes to Dark theme.

» To limit battery use for apps that you don't use often, swipe up from the home screen and tap **Settings** > **Battery** > **Adaptive Battery**, and turn it on.

» To charge more efficiently and keep your battery healthy, swipe up from the home screen and tap **Settings** > **Battery** > **Optimized charging**, and turn it on.

» To protect your battery from being overcharged, swipe up from the home screen and tap **Settings** > **Battery** > **Overcharge protection**, and turn it on.

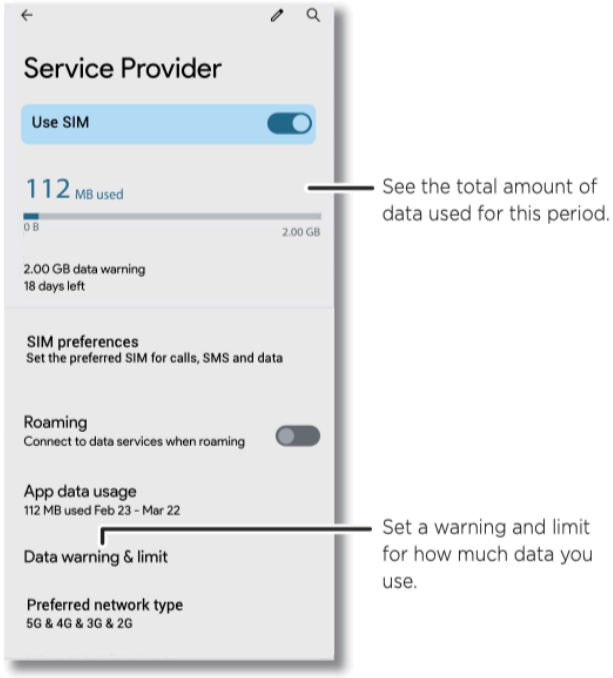
» To show battery percentage in status bar, swipe up from the home screen and tap **Settings** > **Battery**, then tap the switch next to **Battery Percentage** to turn it on.

[1e] and if it is determined that the service usage activity is the background activity, apply the policy.	The Accused Instrumentalities comprise "and if it is determined that the service usage activity is the background activity, apply the policy." For example, Motorola phones and tablets utilize various features (e.g., Data Saver, Power Saver, Adaptive Battery, Doze Mode) which applies the policy to background service usage activity over wireless networks, e.g., when apps utilize network access, jobs, syncs, alarms, etc. *See, e.g.*, https://en-us.support.motorola.com/ci/fattach/get/81413275/1686837226/redirect/1/session/L2F2LzEvdGltZS8xNzEwNzI3MjU3L2dlbi8xNzEwNzI3MjU3L3NpZC9mVWc0dzJqVlJvVU92V2lZJTdFNGNwU3JySkJMa-VJnd2hnMUtYaDdrXzZQYUQ3dVZrSDlmaHNOU05ZSWFWJTdFb2F1Wng4OUpNTEFsWU50WHF1aEw0YWFSOTc2M3NWUGRBeEdPRXVyc1A1UGJQVHZmYXhiWEw2dzg2UHdnJTIxJTIx/filename/motorola+razr%2B+-+2023.NA+Retail.UG.en-US.SSC8D89424-B.pdf:

Data usage

You can track the amount of data your phone uploads and downloads.

Find it: Swipe up from the home screen and tap **Settings** > **Network & internet** > **Mobile network** > choose a SIM card



To see a breakdown of which apps are using data, swipe up from the home screen and tap **Settings** > **Network & internet** > **Mobile network** > **App data usage**. Some apps transfer data in the background when you're not viewing them—to help reduce this type of data usage, swipe up from the home screen and tap **Settings** > **Network & internet** > **Data Saver**, then tap the switch next to **Use Data Saver** to turn it on.

Tip: To see Wi-FI data usage, swipe up from the home screen and tap **Settings** > **Network & internet** > **Internet** > **Non-carrier data usage**.

Note: Usage information is provided to help you manage your phone. This may not match the amounts charged by your service provider, as they're not measured in the same way.

Improve battery life

Your phone processes tons of information. Depending on what apps are in use, your phone may use a lot of power.

When your phone is not in use for a period of time, unnecessary background processes are shut down to optimize battery life.

» To see what's using up battery power, swipe up from the home screen and tap **Settings** > **Battery** > **Battery usage**.

» To help improve battery life, swipe up from the home screen and tap **Settings** > **Battery** > **Battery Saver**, and tap the switch next to **Use Battery Saver** to turn it on. When on, your phone's display changes to Dark theme.

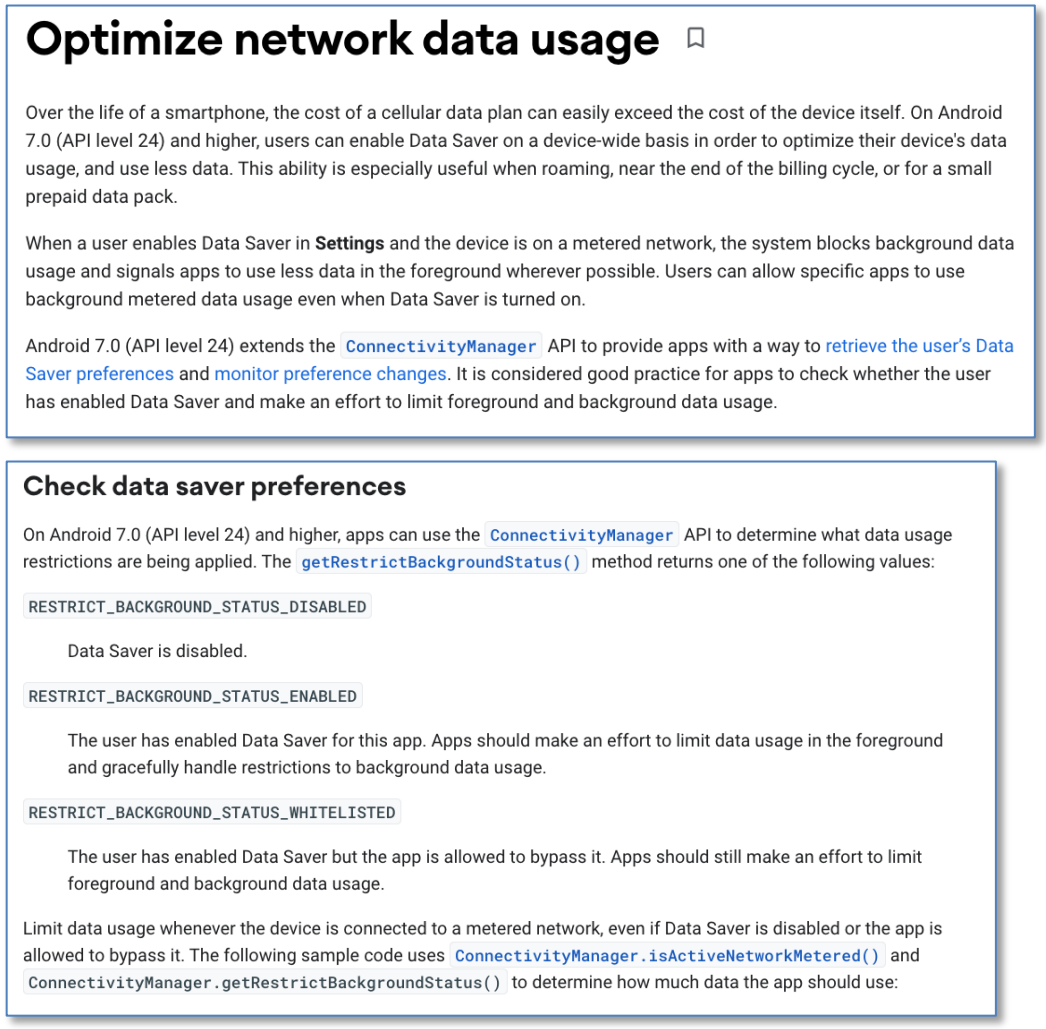
» To limit battery use for apps that you don't use often, swipe up from the home screen and tap **Settings** > **Battery** > **Adaptive Battery**, and turn it on.

» To charge more efficiently and keep your battery healthy, swipe up from the home screen and tap **Settings** > **Battery** > **Optimized charging**, and turn it on.

» To protect your battery from being overcharged, swipe up from the home screen and tap **Settings** > **Battery** > **Overcharge protection**, and turn it on.

» To show battery percentage in status bar, swipe up from the home screen and tap **Settings** > **Battery**, then tap the switch next to **Battery Percentage** to turn it on.

; https://developer.android.com/training/basics/network-ops/data-saver:



Optimize network data usage

Over the life of a smartphone, the cost of a cellular data plan can easily exceed the cost of the device itself. On Android 7.0 (API level 24) and higher, users can enable Data Saver on a device-wide basis in order to optimize their device's data usage, and use less data. This ability is especially useful when roaming, near the end of the billing cycle, or for a small prepaid data pack.

When a user enables Data Saver in **Settings** and the device is on a metered network, the system blocks background data usage and signals apps to use less data in the foreground wherever possible. Users can allow specific apps to use background metered data usage even when Data Saver is turned on.

Android 7.0 (API level 24) extends the `ConnectivityManager` API to provide apps with a way to retrieve the user's Data Saver preferences and monitor preference changes. It is considered good practice for apps to check whether the user has enabled Data Saver and make an effort to limit foreground and background data usage.

Check data saver preferences

On Android 7.0 (API level 24) and higher, apps can use the `ConnectivityManager` API to determine what data usage restrictions are being applied. The `getRestrictBackgroundStatus()` method returns one of the following values:

`RESTRICT_BACKGROUND_STATUS_DISABLED`

Data Saver is disabled.

`RESTRICT_BACKGROUND_STATUS_ENABLED`

The user has enabled Data Saver for this app. Apps should make an effort to limit data usage in the foreground and gracefully handle restrictions to background data usage.

`RESTRICT_BACKGROUND_STATUS_WHITELISTED`

The user has enabled Data Saver but the app is allowed to bypass it. Apps should still make an effort to limit foreground and background data usage.

Limit data usage whenever the device is connected to a metered network, even if Data Saver is disabled or the app is allowed to bypass it. The following sample code uses `ConnectivityManager.isActiveNetworkMetered()` and `ConnectivityManager.getRestrictBackgroundStatus()` to determine how much data the app should use:

; https://developer.android.com/training/monitoring-device-state/doze-standby:

Optimize for Doze and App Standby

Starting from Android 6.0 (API level 23), Android introduces two power-saving features that extend battery life for users by managing how apps behave when a device is not connected to a power source. *Doze* reduces battery consumption by deferring background CPU and network activity for apps when the device is unused for long periods of time. *App Standby* defers background network activity for apps with which the user has not recently interacted.

While the device is in Doze, apps' access to certain battery-intensive resources is deferred until maintenance windows. The specific restrictions are listed in Power Management Restrictions.

Doze and App Standby manage the behavior of all apps running on Android 6.0 or higher, regardless whether they are specifically targeting API level 23. To ensure the best experience for users, test your app in Doze and App Standby modes and make any necessary adjustments to your code. The sections below provide details.

Understanding Doze

If a user leaves a device unplugged and stationary for a period of time, with the screen off, the device enters Doze mode. In Doze mode, the system attempts to conserve battery by restricting apps' access to network and CPU-intensive services. It also prevents apps from accessing the network and defers their jobs, syncs, and standard alarms.

Periodically, the system exits Doze for a brief time to let apps complete their deferred activities. During this *maintenance window*, the system runs all pending syncs, jobs, and alarms, and lets apps access the network.

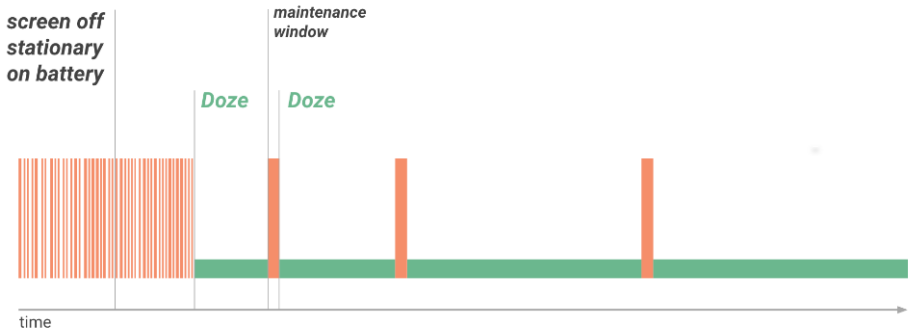


Figure 1. Doze provides a recurring maintenance window for apps to use the network and handle pending activities.

	At the conclusion of each maintenance window, the system again enters Doze, suspending network access and deferring jobs, syncs, and alarms. Over time, the system schedules maintenance windows less and less frequently, helping to reduce battery consumption in cases of longer-term inactivity when the device is not connected to a charger. As soon as the user wakes the device by moving it, turning on the screen, or connecting a charger, the system exits Doze and all apps return to normal activity. The Doze restriction on network access is also likely to affect your app, especially if the app relies on real-time messages such as tickles or notifications. If your app requires a persistent connection to the network to receive messages, you should use Firebase Cloud Messaging (FCM) if possible. ; https://developer.android.com/topic/performance/appstandby:

App Standby Buckets

Android 9 (API level 28) and higher support **App Standby Buckets**. App Standby Buckets help the system prioritize apps' requests for resources based on how recently and how frequently the apps are used. Based on app usage patterns, each app is placed in one of five priority **buckets**. The system limits the device resources available to each app based on which bucket the app is in.

Priority buckets

The system dynamically assigns each app to a priority bucket, reassigning the apps as needed. The system may rely on a preloaded app that uses machine learning to determine how likely each app is to be used, and assigns apps to the appropriate buckets. If the system app is not present on a device, the system defaults to sorting apps based on how recently they were used. More active apps are assigned to buckets that give the apps higher priority, making more system resources available to the app. In particular, the bucket determines how frequently the app's jobs run, and how often the app can trigger alarms. These restrictions apply only while the device is on battery power; the system does not impose these restrictions on apps while the device is charging.

Note: Every manufacturer can set their own criteria for how non-active apps are assigned to buckets. You should not try to influence which bucket your app is assigned to. Instead, focus on making sure your app behaves well in whatever bucket it might be in. Your app can find out what bucket it's currently in by calling `UsageStatsManager.getAppStandbyBucket()`.

The buckets are:

1. Active: App is currently being used or was very recently used.
2. Working set: App is in regular use.
3. Frequent: App is often used, but not every day.
4. Rare: App is not frequently used.
5. Restricted: App consumes a great deal of system resources, or may exhibit undesirable behavior.

In addition, there's a special **never** bucket for apps that have been installed but have never been run. The system imposes severe restrictions on these apps.

	; https://developer.android.com/topic/performance/power/power-details: **Power management restrictions** As described in Power management, the system can impose power restrictions on apps for a number of reasons. The following table outlines the current restrictions. These restrictions do not apply while the device is charging. In each case, the most restrictive applicable setting is the one that takes effect. For example, if Battery Saver is active and an app is in the Rare bucket, the more stringent App Standby Buckets restrictions on Firebase Cloud Messaging (FCM) are applied.

Setting	Jobs *	Alarms	Network †	Firebase Cloud Messaging §
User Restricts Background Activity				
Restrictions enabled:	Never	Never	No restriction	No restriction
Doze				
Doze active:	Deferred to window	Regular alarms: Deferred to window Inexact while-idle alarms: Limited to 1 per 9 minutes Exact while-idle alarms: Limited to 72 per hour	Deferred to window	High priority: No restriction Normal priority: Deferred to window
App Standby Buckets (by bucket)				Prior to Android 13 (API Level 33)
Active:	No restriction	No restriction	No restriction	No restriction
Working set:	Limited to 10 minutes every 2 hours	Limited to 10 per hour	No restriction	No restriction
Frequent:	Limited to 10 minutes every 8 hours	Limited to 2 per hour	No restriction	High priority: 10/day
Rare:	Limited to 10 minutes every 24 hours	Limited to 1 per hour	Disabled	High priority: 5/day
Restricted:	Once per day	One alarm per day, either an exact alarm or an inexact alarm	Disabled	High priority: 5/day

	; https://developer.android.com/topic/performance/background-optimization; https://developer.android.com/reference/android/app/job/JobScheduler; https://developer.android.com/guide/background/persistent; https://developer.android.com/guide/components/activities/activity-lifecycle:

Activity-lifecycle concepts

To navigate transitions between stages of the activity lifecycle, the `Activity` class provides a core set of six callbacks: `onCreate()`, `onStart()`, `onResume()`, `onPause()`, `onStop()`, and `onDestroy()`. The system invokes each of these callbacks as the activity enters a new state.

Figure 1 presents a visual representation of this paradigm.

As the user begins to leave the activity, the system calls methods to dismantle the activity. In some cases, the activity is only partially dismantled and still resides in memory, such as when the user switches to another app. In these cases, the activity can still come back to the foreground.

If the user returns to the activity, it resumes from where the user left off. With a few exceptions, apps are restricted from starting activities when running in the background.

The system's likelihood of killing a given process, along with the activities in it, depends on the state of the activity at the time. For more information on the relationship between state and vulnerability to ejection, see the section about activity state and ejection from memory.

Depending on the complexity of your activity, you probably don't need to implement all the lifecycle methods. However, it's important that you understand each one and implement those that make your app behave the way users expect.

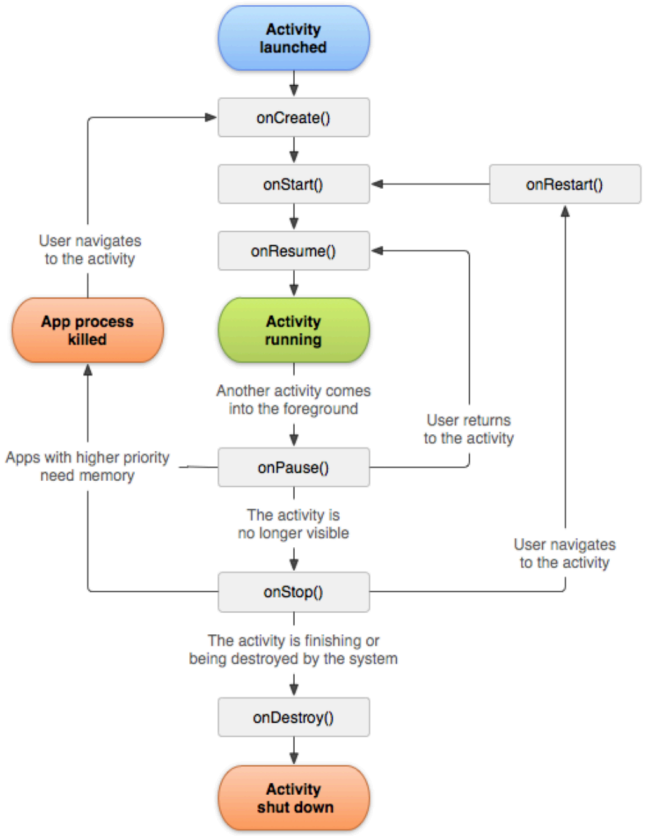


Figure 1. A simplified illustration of the activity lifecycle.